CHAPTER

2

# Data Set Options

## Definition

*Data set options* specify actions that apply only to the SAS data set with which they appear. They let you perform such operations as

- renaming variables
- selecting only the first or last *n* observations for processing
- dropping variables from processing or from the output data set
- specifying a password for a SAS data set.

## Syntax

Specify a data set option in parentheses after a SAS data set name. To specify several data set options, separate them with spaces.

(*option-1=value-1<. . . option-n=value-n>*)

These examples show data set options in SAS statements:

- `data scores(keep=team game1 game2 game3);`
- `proc print data=new(drop=year);`
- `set old(rename=(date=Start_Date));`

## Using Data Set Options

### Using Data Set Options with Input or Output SAS Data Sets

Most SAS data set options can apply to either input or output SAS data sets in DATA steps or procedure (PROC) steps. If a data set option is associated with an input data set, the action applies to the data set that is being read. If the option appears in the DATA statement or after an output data set specification in a PROC step, SAS applies the action to the output data set. In the DATA step, data set options for output data sets must appear in the DATA statement, not in any OUTPUT statements that may be present.

Some data set options, such as COMPRESS=, are meaningful only when you create a SAS data set because they set attributes that exist for the life of the data set. To change or cancel most data set options, you must re-create the data set. You can change other options (such as PW= and LABEL=) with PROC DATASETS. For more information, see “The DATASETS Procedure” in the *SAS Procedures Guide*.

When data set options appear on both input and output data sets in the same DATA or PROC step, SAS applies data set options to input data sets before it evaluates programming statements or before it applies data set options to output data sets.

In some instances, data set options conflict when they are used in the same statement. For example, you cannot specify both the DROP= and KEEP= options for the same variable in the same statement.

## How Data Set Options Interact with System Options

Many system options and data set options share the same name and have the same function. System options remain in effect for all DATA and PROC steps in a SAS job or session unless they are respecified.

The data set option overrides the system option for the data set in the step in which it appears. In this example, the OBS= system option in the OPTIONS statement specifies that only the first 100 observations will be processed from any data set within the SAS job. The OBS= data set option in the SET statement, however, overrides the system option for data set TWO and specifies that only the first 5 observations will be read from data set TWO. The PROC PRINT step prints the data set FINAL. This data set contains the first 5 observations from data set TWO, followed by the first 100 observations from data set THREE:

```
options obs=100;

data final;
   set two(obs=5) three;
run;

proc print data=final;
run;
```

## Data Set Options by Category

**Table 2.1** Categories and Descriptions of Data Set Options

| Category | Data Set Option | Description |
|---|---|---|
| Data Set Control | “ALTER=” on page 9 | Assigns an alter password to a SAS file and enables access to a password-protected SAS file |
| | “BUFNO=” on page 9 | Specifies the number of buffers for processing a SAS data set |
| | “BUFSIZE=” on page 10 | Specifies a permanent buffer size for output SAS data sets |
| | “CNTLLEV=” on page 11 | Specifies the level of shared access to SAS data sets |
| | “COMPRESS= ”on page 12 | Compresses observations in an output SAS data set |
| | “DLDMGACTION=” on page 14 | Specifies what type of action to take when a SAS data set in a SAS data library is detected as damaged |
| | “ENCRYPT=” on page 16 | Encrypts SAS data files |
| | “GENMAX=” on page 19 | Requests generations for a data set and specifies the maximum number of versions |
| | “GENNUM=” on page 20 | References a specific generation of a data set |
| | “INDEX=” on page 26 | Defines indexes when a SAS data set is created |
| | “LABEL=” on page 28 | Specifies a label for the data set |

| | | |
|---|---|---|
| | "OUTREP=" on page 31 | Specifies an operating environment's requirements vector for an output file |
| | "PW=" on page 32 | Assigns a read, write, or alter password to a SAS file and enables access to a password-protected SAS file |
| | "PWREQ=" on page 33 | Controls the pop up of a requestor window for a data set password |
| | "READ=" on page 34 | Assigns a read password to a SAS file and enables access to a read-protected SAS file |
| | "REPEMPTY=" on page 36 | Controls replacement of like-named temporary or permanent SAS data sets when the new one is empty. |
| | "REPLACE=" on page 37 | Controls replacement of like-named temporary or permanent SAS data sets |
| | "REUSE=" on page 38 | Specifies whether new observations are written to free space in compressed SAS data sets |
| | "SORTEDBY=" on page 39 | Specifies how the data set is currently sorted |
| | "TOBSNO=" on page 41 | Specifies the number of observations to be transmitted in each multi-observation exchange with a SAS server |
| | "TRANTAB=" on page 42 | Specifies a translation table for character conversions |
| | "TYPE=" on page 42 | Specifies the data set type for a specially structured SAS data set |
| | "WRITE=" on page 46 | Assigns a write password to a SAS file and enables access to a write-protected SAS file |
| Miscellaneous | "FILECLOSE=" on page 17 | Specifies how a tape is positioned when a SAS file on the tape is closed |
| Observation Control | "FIRSTOBS=" on page 18 | Causes processing to begin at a specified observation |
| | "IN=" on page 24 | Creates a variable that indicates whether the data set contributed data to the current observation |
| | "OBS=" on page 29 | Causes processing to end with the *n*th observation |
| | "POINTOBS= "on page 31 | Controls whether a compressed data set may be processed with random access (by observation number) rather than sequential access only |
| | "WHERE=" on page 43 | Selects observations that meet the specified condition |
| | "WHEREUP=" on page 45 | Specifies whether to evaluate added observations and modified observations against a WHERE expression |
| User Control of SAS Index Usage | "IDXNAME=" on page 21 | Directs SAS to use a specific index to satisfy the conditions of a WHERE expression |
| | "IDXWHERE=" on page 23 | Overrides the SAS System decision about whether to use an index to satisfy the conditions of a WHERE expression |
| Variable Control | "DROP=" on page 15 | Excludes variables from processing or from output SAS data sets |

| | |
|---|---|
| "KEEP=" on page 27 | Specifies variables for processing or for writing to output SAS data sets |
| "RENAME=" on page 35 | Changes the name of a variable |

## Dictionary

## ALTER=

**Assigns an alter password to a SAS file and enables access to a password-protected SAS file**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

### Syntax

ALTER=*alter-password*

### Syntax Description

***alter-password***
must be a SAS name.

### Details

The ALTER= option applies to all types of SAS files except catalogs. You can use this option to assign an *alter-password* to a SAS file or to access a read-, write-, or alter-protected SAS file.

### See Also

Data Set Options:
- "ENCRYPT=" on page 16
- "PW=" on page 32
- "READ=" on page 34
- "WRITE=" on page 46

"File Protection" in *SAS Language Reference: Concepts*

"Manipulating Passwords" in "The DATASETS Procedure" in the *SAS Procedures Guide*

## BUFNO=

**Specifies the number of buffers for processing a SAS data set**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

## Syntax

BUFNO=*number-of-buffers*

## Syntax Description

***number-of-buffers***
specifies a value from 1 to the maximum number of buffers available in your operating environment.

## Details

The buffer number is not a permanent attribute of the data set and is valid only for the current SAS session or job.

*Note:* To reduce input/output operations on a small data set as well as speed execution time, allocate one buffer for each page of data to be processed. This technique is most effective if you read the same observations several times during processing. △

## Comparisons

If the BUFNO= data set option is not specified, then the value of the BUFNO= system option is used. If both are specified in the same SAS session, the value specified for the BUFNO= data set option overrides the value specified for the BUFNO= system option.

## See Also

Data Set Options:
“BUFSIZE=” on page 10
System Options:
“BUFNO=” on page 1063

## BUFSIZE=

**Specifies a permanent buffer size for output SAS data sets**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

**Restriction:** Use with output data sets only.

## Syntax

BUFSIZE=*number-of-bytes*

## Syntax Description

***number-of-bytes***
specifies the minimum number of bytes in a page in the data set. The number of bytes can have a value from 0, the system default, to the maximum value allowed by your operating environment. If the value is 0, SAS chooses an operating environment default that is optimal for the SAS data set. If any value you specify is not adequate, SAS automatically rounds up to the next buffer size for the data set.

## Details

The BUFSIZE= data set option is valid only for output data sets, that is, data sets that are created in a DATA step or by a SAS procedure.

The buffer size, or page size, determines the size of a single input/output buffer that SAS uses to transfer data during processing. A *page* is the minimum number of bytes of data that SAS moves between external storage and memory in one logical input/output operation. Once it is specified, the buffer size is a permanent attribute of the data set, and the specified buffer size is used whenever the data set is processed. To change the buffer size, use a DATA step to copy the data set and either specify a new buffer size or use the SAS default.

*Note:* If you use the COPY procedure to copy a data set to another library that is allocated with a different engine, the specified buffer size of the data set is not retained. △

*Operating Environment Information:* For details, see the SAS documentation for your operating environment for minimum and maximum buffer size values. △

*Note:* Using the BUFSIZE= option can speed up execution time by reducing the number of times SAS has to read from or write to the storage medium. However, the improvement in execution time comes at the cost of increased memory consumption. △

## Comparisons

If the BUFSIZE= data set option is not specified, then the value of the BUFSIZE= system option is used. If both are specified in the same SAS session, the BUFSIZE= data set option overrides the value specified for the BUFSIZE= system option.

## See Also

Data Set Options:
    "BUFNO=" on page 9
System Options:
    "BUFSIZE=" on page 1064

## CNTLLEV=

**Specifies the level of shared access to SAS data sets**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

**Restriction:** Specify for input data sets only.

### Syntax

CNTLLEV=LIB | MEM | REC

### Syntax Description

**LIB**
controls concurrent access at the library level. Library-level control restricts concurrent access to only one update process to the library.

**MEM**
controls concurrent access at the SAS data set (or member) level. Member-level control restricts concurrent access to only one update or output process but allows read access to many sessions, procedures, or statements.

**REC**
controls concurrent access at the observation (or record) level. Record-level control allows more than one update access to the same SAS data set, but it denies concurrent update of the same observation.

### Details

The CNTLLEV= option specifies the level at which shared update access to a SAS data set is denied. A SAS data set can be opened concurrently by more than one SAS session or by more than one statement, window, or procedure within a single session. By default, SAS procedures permit the greatest degree of concurrent access possible while they guarantee the integrity of the data and the data analysis. Therefore, you do not normally use the CNTLLEV= option.

Use this option when

- □ your application controls the access to the data, such as in SAS Component Language (SCL), SAS/IML software, or DATA step programming
- □ you access data through an interface engine that does not provide member-level control of the data.

If you use CNTLLEV=REC and the SAS procedure needs member-level control for integrity of the data analysis, SAS prints a warning to the SAS log that inaccurate or unpredictable results can occur if the data are updated by another process during the analysis.

## COMPRESS=

**Compresses observations in an output SAS data set**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

**Restriction:** Use with output data sets only.

## Syntax

COMPRESS= YES | NO | CHAR | BINARY

## Syntax Description

**YES | CHAR**
specifies that the observations in a newly created SAS output data set are compressed (variable-length records).

**Tip:** SAS uses RLE (Run Length Encoding) to compress observations. This compression algorithm is better for character data.

**NO**
specifies that the observations in a newly created SAS data set are uncompressed (fixed-length records).

**BINARY**
specifies that observations in a newly created SAS output data set are compressed.

**Tip:** SAS uses RDC (Ross Data Compression) for this setting. This method is highly effective for compressing medium to large (several hundred bytes or larger) blocks of binary data (that is, numeric variables). Because the compression function operates on a single record at a time, the record length needs to be several hundred bytes or larger for effective compression.

## Details

Specify COMPRESS= only for output data sets, that is, data sets named in the DATA statement of a DATA step or in the OUT= option of a SAS procedure. The record type becomes a permanent attribute of the data set. To uncompress observations, use a DATA step to copy the data set and use COMPRESS=NO for the new data set.

When COMPRESS=YES|CHAR, SAS compresses the size of the data set with run-length encoding. Run-length encoding compresses the data set by reducing repeated consecutive characters to two- or three-byte representations. When COMPRESS=BINARY, SAS compression combines run-length encoding and sliding-window compression to compress the data set.

Use SAS/Toolkit to specify your own compression method.

*Note:* Compression of observations is not supported by all engines. △

In Version 8, data sets created with engines that were available in earlier versions of SAS, such as the TAPE and XPORT engines, are still accessed by those engines. Therefore, if compression was unavailable for those engines, it is also not available when you access those data sets in Version 8.

The advantages gained by using the COMPRESS= data set option include:

- reduced storage requirements for the data set
- fewer input and output operations necessary to read from or write to the data set during processing.

The disadvantages of using the COMPRESS= data set option include:

- may not compress at all (may actually make the file larger), but a message detailing the amount of compression is provided
- more CPU resources are required.

By default, new observations are appended to existing compressed data sets. If you want to track and reuse free space, use the REUSE= data set option when you create a compressed SAS data set. REUSE=YES tells SAS to write new observations to the space that is freed when you delete other observations.

### Comparisons

The COMPRESS= data set option overrides the COMPRESS= system option.

PERFORMANCE NOTE: Using this option increases the CPU time for reading a data set because of the overhead of uncompressing the record. In addition, some engines do not support compression of observations. When using COMPRESS=YES and REUSE=YES option settings, observations cannot be addressed by observation number.

Note that REUSE=YES takes precedence over POINTOBS=YES. For example:

```
data test(compress=yes pointobs=yes reuse=yes);
```

results in a data set that has POINTOBS=NO. Because POINTOBS=YES is the default when you use compression, REUSE=YES causes POINTOBS= to change to NO.

### See Also

Data Set Options:

- “POINTOBS= ”on page 31
- “REUSE=” on page 38

System Options:

- “COMPRESS=” on page 1077
- “REUSE=” on page 1143

## DLDMGACTION=

**Specifies what type of action to take when a SAS data set in a SAS data library is detected as damaged**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

### Syntax

DLDMGACTION=FAIL | ABORT | REPAIR | PROMPT

### Syntax Description

**FAIL**
stops the step, issues an error message to the log immediately. This is the default for batch mode.

**ABORT**
terminates the step, issues an error message to the log, and aborts the SAS session.

**REPAIR**
automatically repairs and rebuilds indexes and integrity constraints. It issues a warning message to the log. This is the default for interactive mode.

**PROMPT**
displays a requestor window that asks you to select the FAIL, ABORT, or REPAIR action.

## DROP=

**Excludes variables from processing or from output SAS data sets**

**Valid in:** DATA step and PROC steps

**Category:** Variable Control

### Syntax

DROP=*variable(s)*

### Syntax Description

***variable(s)***
lists one or more variable names. You can list the variables in any form that SAS allows.

### Details

If the option is associated with an input data set, the variables are not available for processing. If the DROP= data set option is associated with an output data set, SAS does not write the variables to the output data set, but they are available for processing.

### Comparisons

- The DROP= data set option differs from the DROP statement in these ways:
    - In DATA steps, the DROP= data set option can apply to both input and output data sets. The DROP statement applies only to output data sets.
    - In DATA steps, when you create multiple output data sets, use the DROP= data set option to write different variables to different data sets. The DROP statement applies to all output data sets.
    - In PROC steps, you can use only the DROP= data set option, not the DROP statement.
- The KEEP= data set option specifies a list of variables to be included in processing or to be written to the output data set.

### Examples

**Example 1: Excluding Variables from Input** In this example, the variables SALARY and GENDER are not included in processing and they are not written to either output data set:

```
data plan1 plan2;
   set payroll(drop=salary gender);
   if hired<'01jan98'd then output plan1;
   else output plan2;
run;
```

You cannot use SALARY or GENDER in any logic in the DATA step because DROP= prevents the SET statement from reading them from PAYROLL.

**Example 2: Processing Variables without Writing Them to a Data Set** In this example, SALARY and GENDER are not written to PLAN2, but they are written to PLAN1:

```
data plan1 plan2(drop=salary gender);
   set payroll;
   if hired<'01jan98'd then output plan1;
   else output plan2;
run;
```

## See Also

Data Set Options:
  "KEEP=" on page 27
Statements:
  "DROP" on page 796

## ENCRYPT=

**Encrypts SAS data files**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

**Restriction:** Use with output data sets only.

### Syntax

ENCRYPT=YES | NO

### Syntax Description

**YES**
encrypts the file. The encryption method uses passwords. At a minimum, you must specify the READ= or the PW= data set option at the same time that you specify ENCRYPT=YES. Because the encryption method uses passwords, you cannot change *any* password on an encrypted data set without re-creating the data set.

**NO**
does not encrypt the file.

***CAUTION:***

**Record all passwords.** If you forget the password, you cannot reset it without assistance from SAS Institute. The process is time-consuming and resource-intensive. △

## Details

- You can use the ENCRYPT= option only when you are creating a SAS data file.
- In order to copy an encrypted SAS data file, the output engine must support encryption. Otherwise, the data file is not copied.
- Encrypted files work only in Release 6.11 or in later releases of SAS.
- You cannot encrypt SAS data views or stored programs because they contain no data.
- If the data file is encrypted, all associated indexes are also encrypted.
- Encryption requires roughly the same amount of CPU resources as compression.
- You cannot use PROC CPORT on encrypted SAS data files.

## Example

This example creates an encrypted SAS data set:

```
data salary(encrypt=yes read=green);
   input name $ yrsal bonuspct;
   datalines;
Muriel    34567  3.2
Bjorn     74644  2.5
Freda     38755  4.1
Benny     29855  3.5
Agnetha   70998  4.1
;
```

To use this data set, specify the read password:

```
proc contents data=salary(read=green);
run;
```

## See Also

Data Set Options:

“ALTER=” on page 9
“PW=” on page 32
“READ=” on page 34
“WRITE=” on page 46

“SAS File Encryption” in *SAS Language Reference: Concepts*

---

## FILECLOSE=

**Specifies how a tape is positioned when a SAS file on the tape is closed**

**Valid in:** DATA step and PROC steps

**Category:** Miscellaneous

## Syntax

FILECLOSE=DISP | LEAVE | REREAD | REWIND

## Syntax Description

**DISP**
positions the tape volume according to the disposition specified in the operating environment's control language.

**LEAVE**
positions the tape at the end of the file that was just processed. Use FILECLOSE=LEAVE if you are not repeatedly accessing the same files in a SAS program but you are accessing one or more subsequent SAS files on the same tape.

**REREAD**
positions the tape volume at the beginning of the file that was just processed. Use FILECLOSE=REREAD if you are accessing the same SAS data set on tape several times in a SAS program.

**REWIND**
rewinds the tape volume to the beginning. Use FILECLOSE=REWIND if you are accessing one or more previous SAS files on the same tape, but you are not repeatedly accessing the same files in a SAS program.

*Operating Environment Information:* These values are not recognized by all operating environments. Additional values are available on some operating environments. See the appropriate sections of the SAS documentation for your operating environment for more information on using SAS data libraries that are stored on tape. △

## FIRSTOBS=

**Causes processing to begin at a specified observation**

**Valid in:** DATA step and PROC steps

**Category:** Observation Control

**Restriction:** Use with input data sets only.

### Syntax

FIRSTOBS=*n*

### Syntax Description

***n***
is a positive integer that is less than or equal to the number of observations in the data set.

### Details

The FIRSTOBS= data set option is valid when an existing SAS data set is read. You cannot use this option when a WHERE statement or WHERE= data set option is specified in the same DATA or PROC step.

### Comparisons

- □ The FIRSTOBS= data set option overrides the FIRSTOBS= system option for the individual data set.
- □ While the FIRSTOBS= data set option specifies a starting point for processing, the OBS= data set option specifies an ending point. The two options are often used together to define a range of observations to be processed.
- □ When external files are read, the FIRSTOBS= option in the INFILE statement specifies which record to read first.

### Examples

This PROC step prints the data set STUDY beginning with observation 20:

```
proc print data=study(firstobs=20);
run;
```

This SET statement uses both FIRSTOBS= and OBS= to read only observations 5 through 10 from the data set STUDY. Data set NEW contains six observations.

```
data new;
  set study(firstobs=5 obs=10);
run;
```

### See Also

Data Set Options:
“OBS=” on page 29

Statements:
“INFILE” on page 857
“WHERE” on page 1028

System Options:
“FIRSTOBS=” on page 1098

## GENMAX=

**Requests generations for a data set and specifies the maximum number of versions**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

### Syntax

GENMAX=*number-of-generations*

### Syntax Description

***number-of-generations***
requests generations for a data set and specifies the maximum number of versions to maintain. The value can be from 0 to 1000. The default is GENMAX=0, which means that generations is not in effect.

### Details

You use GENMAX= to request generations for a new data set and to modify the number of generations on an existing data set. The first time the data set is replaced, SAS keeps the replaced version and appends a four-character version number to its member name, which includes # and a three-digit number. For example, for a data set named A, a historical version would be A#001.

Once generations is requested, its member name is limited to 28 characters (rather than 32), because the last four characters are reserved for the appended version number. When generations is not in effect (that is, GENMAX=0), the member name can be up to 32 characters.

If you reduce the number of generations on an existing data set, SAS deletes the oldest version(s) above the new limit.

For more details on generation data sets, see "SAS Data Sets" in *SAS Language Reference: Concepts*.

### Examples

**Example 1: Requesting Generations When You Create a Data Set** This example shows how to request generations for a new data set. The DATA step creates a data set namedWORK.A that can have as many as 10 generations (one current version and nine historical versions):

```
data a(genmax=10);
   x=1;
   output;
run;
```

**Example 2: Modifying the Number of Generations on an Existing Data Set** This example shows how to change the number of generations on the data set MYLIB.A to 4:

```
proc datasets lib=mylib;
   modify a(genmax=4);
run;
```

### See Also

Data Set Option:
"GENNUM=" on page 20

## GENNUM=

**References a specific generation of a data set**

**Valid in:** DATA step and PROC steps
**Category:** Data Set Control

## Syntax

GENNUM=*integer*

## Syntax Description

***integer***
is a number that references a specific version from a generation group. Specifying a positive number is an absolute reference to a specific version number that is appended to a data set's name. Specifying a negative number is a relative reference to a version in relation to the base version, from the youngest to the oldest. A value of 0 refers to the current (base) version. Specify an asterisk for *integer* when you rename the entire generation group for a data set.

## Details

After generations for a data set have been requested using the GENMAX= data set option, use GENNUM= to request a specific version. For example, specifying GENNUM=3 refers to the historical version #003, while specifying GENNUM=-1 refers to the youngest historical version.

Note that after 999 replacements, the youngest version would be #999. After 1,000 replacements, SAS rolls over the youngest version number to #000. Therefore, if you want the historical version #000, specify GENNUM=1000.

For more details on generation data sets, see "SAS Data Sets" in *SAS Language Reference: Concepts*.

## Examples

**Example 1: Requesting a Version Using an Absolute Reference** This example prints the historical version #003 for data set A, using an absolute reference:

```
proc print data=a(gennum=3);
run;
```

**Example 2: Requesting A Version Using a Relative Reference** The following PRINT procedure prints the data set three versions back from the base version:

```
proc print data=a(gennum-3);
run;
```

## See Also

Data Set Option:
"GENMAX=" on page 19

# IDXNAME=

**Directs SAS to use a specific index to satisfy the conditions of a WHERE expression**

**Valid in:** DATA step and PROC steps

**Category:** User Control of SAS Index Usage

**Restriction:** Use with input data sets only.

**Restriction:** Mutually exclusive with IDXWHERE= data set option

## Syntax

IDXNAME=*index-name*

## Syntax Description

***index-name***
specifies the name (up to 32 characters) of a simple or composite index for the SAS data set. SAS does not attempt to determine if the specified index is the best one or if a sequential search might be more resource-efficient.

## Details

By default, to satisfy the conditions of a WHERE expression for an indexed SAS data set, SAS decides whether to use an index or read the data set sequentially. The software estimates the relative efficiency and chooses the method that is more efficient.

You might need to direct the software to use a specific index by specifying the IDXNAME= data set option because the decision is based on general rules that may occasionally not produce the best results. That is, by specifying the index, you are able to control which index is used.

The SAS System uses the specified index if the following restrictions are true:

- □ The specified index must exist.
- □ The specified index must be suitable by having at least its first or only variable match a condition in the WHERE expression.
- □ The specified index cannot conflict with BY processing requirements. That is, if a BY statement is included, the specified index must be usable for both the BY statement and the WHERE expression.
- □ The specified index cannot conflict with missing value requirements. That is, if the specified index is created with the NOMISS option so that missing values are not maintained in the index, the WHERE expression cannot qualify any observations that contain missing values. For example, suppose the index AGE was created with the NOMISS option and observations exist that contain missing values for the variable AGE. You cannot specify the following:

```
data mydata.empnew;
   set mydata.employee (idxname=age);
   where age < 35;
```

*Note:* The specification is not a permanent attribute of the data set and is valid only for the current use of the data set. △

*Note:* If you issue the system option MSGLEVEL=I, you can request that IDXNAME= usage be noted in the SAS log if the setting affects index processing. △

## Comparisons

IDXWHERE= enables you to override the SAS System decision about whether to use an index.

## Examples

**Example 1: Using a Specific Index** This example uses the IDXNAME= data set option to direct SAS to use a specific index to optimize the WHERE expression. SAS then disregards the possibility that a sequential search of the data set might be more resource-efficient and does not attempt to determine if the specified index is the best one. (Note that the EMPNUM index was not created with the NOMISS option.)

```
data mydata.empnew;
   set mydata.employee (idxname=empnum);
   where empnum < 2000;
```

## See Also

Data Set Option:

“IDXWHERE=” on page 23

“SAS Indexes” in the “SAS Data Files” section of *SAS Language Reference: Concepts*

“WHERE Processing” in *SAS Language Reference: Concepts*

## IDXWHERE=

**Overrides the SAS System decision about whether to use an index to satisfy the conditions of a WHERE expression**

**Valid in:** DATA step and PROC steps

**Category:** User Control of SAS Index Usage

**Restriction:** Use with input data sets only.

**Restriction:** Mutually exclusive with IDXNAME= data set option

### Syntax

IDXWHERE=YES | NO

### Syntax Description

**YES**
tells SAS to choose the best index to optimize a WHERE expression, and to disregard the possibility that a sequential search of the data set might be more resource-efficient.

**NO**
tells SAS to ignore all indexes and satisfy the conditions of a WHERE expression with a sequential search of the data set.

*Note:* You cannot use IDXWHERE= to override the use of an index to process a BY statement. △

## Details

By default, to satisfy the conditions of a WHERE expression for an indexed SAS data set, SAS decides whether to use an index or to read the data set sequentially. The software estimates the relative efficiency and chooses the method that is more efficient.

You might need to override the software's decision by specifying the IDXWHERE= data set option because the decision is based on general rules that may occasionally not produce the best results. That is, by specifying the IDXWHERE= data set option, you are able to determine the processing method.

*Note:* The specification is not a permanent attribute of the data set and is valid only for the current use of the data set. △

*Note:* If you issue the system option MSGLEVEL=I, you can request that IDXWHERE= usage be noted in the SAS log if the setting affects index processing. △

## Comparisons

IDXNAME= enables you to direct SAS to use a specific index.

## Examples

**Example 1: Specifying Index Usage** This example uses the IDXWHERE= data set option to tell SAS to decide which index is the best to optimize the WHERE expression. SAS then disregards the possibility that a sequential search of the data set might be more resource-efficient:

```
data mydata.empnew;
   set mydata.employee (idxwhere=yes);
   where empnum < 2000;
```

**Example 2: Specifying No Index Usage** This examples uses the IDXWHERE= data set option to tell SAS to ignore any index and to satisfy the conditions of the WHERE expression with a sequential search of the data set:

```
data mydata.empnew;
   set mydata.employee (idxwhere=no);
   where empnum < 2000;
```

## See Also

Data Set Option:

"IDXNAME=" on page 21

"SAS Indexes" in the "SAS Data Files" section of *SAS Language Reference: Concepts*

"WHERE Processing" in *SAS Language Reference: Concepts*

## IN=

**Creates a variable that indicates whether the data set contributed data to the current observation**

**Valid in:** DATA step

**Category:** Observation Control

**Restriction:** Use with the SET, MERGE, MODIFY, and UPDATE statements only.

## Syntax

IN=*variable*

## Syntax Description

***variable***
names the new variable whose value indicates whether that input data set contributed data to the current observation. Within the DATA step, the value of the variable is 1 if the data set contributed to the current observation, and 0 otherwise.

## Details

Specify the IN= data set option in parentheses after a SAS data set name in the SET, MERGE, MODIFY and UPDATE statements only. Values of IN= variables are available to program statements during the DATA step, but the variables are not included in the SAS data set that is being created, unless they are explicity assigned to a new variable.

When you use IN= BY–group processing and when a data set contributes an observation for the current BY group, the IN= value is 1. The value remains as long as that BY group is still being processed and the value is not reset by programming logic.

## Examples

In this example, IN= creates a new variable, OVERSEAS, that denotes international flights. The variable I has a value of 1 when the observation is read from the NONUSA data set; otherwise, it has a value of 0. The IF-THEN statement checks the value of I to determine if the data set NONUSA contributed data to the current observation. If I=1, the variable OVERSEAS receives an asterisk (*) as a value.

```
data allflts;
   set usa nonusa(in=i);
   by fltnum;
   if i then overseas='*';
run;
```

## See Also

Statements:

"BY" on page 765

"MERGE" on page 930

"MODIFY" on page 933

"SET" on page 1010

"UPDATE" on page 1023

"BY-Group Processing" in *SAS Language Reference: Concepts*

---

## INDEX=

**Defines indexes when a SAS data set is created**

**Valid in:** DATA step and PROC step

**Category:** Data Set Control

**Restriction:** Use with output data sets only.

### Syntax

INDEX=(*index-specification-1* ...<*index-specification-n*>)

### Syntax Description

***index-specification***

names and describes a simple or a composite index to be built. *Index-specification* has this form:

*index* <= (*variable(s)*) > </UNIQUE> </NOMISS>

| | |
|---|---|
| *index* | is the name of a variable that forms the index or the name you choose for a composite index. |
| *variable(s)* | is a list of variables to use in making a composite index. |
| UNIQUE | specifies that the values of the key variables must be unique. If you specify UNIQUE for a new data set and multiple observations have the same values for the index variables, the index is not created. A slash (/) must precede the UNIQUE option. |
| NOMISS | excludes all observations with missing values from the index. Observations with missing values are still read from the data set but not through the index. A slash (/) must precede the NOMISS option. |

### Examples

**Example 1: Defining a Simple Index** The following INDEX= data set option defines a simple index for the SSN variable:

```
data new(index=(ssn));
```

**Example 2: Defining a Composite Index** The following INDEX= data set option defines a composite index named CITYST that uses the CITY and STATE variables:

```
data new(index=(cityst=(city state)));
```

**Example 3: Defining a Simple and a Composite Index** The following INDEX= data set option defines a simple index for SSN and a composite index for CITY and STATE:

```
data new(index=(ssn cityst=(city state)));
```

### See Also

INDEX CREATE statement in “The DATASETS Procedure” in the *SAS Procedures Guide*

CREATE INDEX statement in “The SQL Procedure” in the *SAS Procedures Guide*

“SAS Indexes” in the “SAS Data Files” section of *SAS Language Reference: Concepts*

## KEEP=

**Specifies variables for processing or for writing to output SAS data sets**

**Valid in:** DATA step and PROC steps

**Category:** Variable Control

### Syntax

KEEP=*variable(s)*

### Syntax Description

***variable(s)***
lists one or more variable names. You can list the variables in any form that SAS allows.

### Details

If the KEEP= data set option is associated with an input data set, only those variables that are listed after the KEEP= data set option are available for processing. If the KEEP= data set option is associated with an output data set, only the variables listed after the option are written to the output data set, but all variables are available for processing.

### Comparisons

- The KEEP= data set option differs from the KEEP statement in the following ways:

  - □ In DATA steps, the KEEP= data set option can apply to both input and output data sets. The KEEP statement applies only to output data sets.
  - □ In DATA steps, when you create multiple output data sets, use the KEEP= data set option to write different variables to different data sets. The KEEP statement applies to all output data sets.
  - □ In PROC steps, you can use only the KEEP= data set option, not the KEEP statement.
- □ The DROP= data set option specifies variables to omit during processing or to omit from the output data set.

### Examples

In this example, only IDNUM and SALARY are read from PAYROLL, and they are the only variables in PAYROLL that are available for processing:

```
data bonus;
   set payroll(keep=idnum salary);
   bonus=salary*1.1;
run;
```

### See Also

Data Set Options:
"KEEP=" on page 27
Statements:
"KEEP" on page 905

## LABEL=

**Specifies a label for the data set**

**Valid in:** DATA step and PROC steps
**Category:** Data Set Control

### Syntax

LABEL=*'label'*

### Syntax Description

***'label'***
is a text string of up to 256 characters. If the label text contains single quotation marks, use double quotation marks around the label, or use two single quotation marks in the label text and surround the string with single quotation marks. To remove a label from a data set, assign a label that is equal to a blank that is enclosed in quotation marks.

## Details

You can use the LABEL= option on both input and output data sets. When you use LABEL= on input data sets, it assigns a label for the file for the duration of that DATA or PROC step. When it is specified for an output data set, the label becomes a permanent part of that file and can be printed using the CONTENTS or DATASETS procedure, and modified using PROC DATASETS.

A label assigned to a data set remains associated with that data set when you update a data set in place, such as when you use the APPEND procedure or the MODIFY statement. However, a label is lost if you use a data set with a previously assigned label to create a new data set in the DATA step. For example, a label previously assigned to data set ONE is lost when you create the new output data set ONE in this DATA step:

```
data one;
     set one;
run;
```

## Comparisons

- □ The LABEL= data set option enables you to specify labels only for data sets. You can specify labels for the variables in a data set using the LABEL statement.
- □ The LABEL= option in the ATTRIB statement also enables you to assign labels to variables.

## Examples

These examples assign labels to data sets:

- □ **`data w2(label='1976 W2 Info, Hourly');`**
- □ **`data new(label='Peter''s List');`**
- □ **`data new(label="Hillside's Daily Account");`**
- □ **`data sales(label='Sales For May(NE)');`**

## See Also

Statements:

“ATTRIB” on page 762

“LABEL” on page 906

“MODIFY” on page 933

“The CONTENTS Procedure” in the *SAS Procedures Guide*

“The DATASETS Procedure” in the *SAS Procedures Guide*

---

## OBS=

**Causes processing to end with the *n*th observation**

**Valid in:** DATA step and PROC steps

**Category:** Observation Control

**Restriction:** Use with input data sets only.

## Syntax

OBS=*n* | MAX

## Syntax Description

***n***
specifies a positive integer that is less than or equal to the number of observations in the data set or zero.

**MAX**
represents the total number of observations in the data set.

## Details

This option specifies the number of the last observation to process, not how many observations should be processed. It is valid only when an existing SAS data set is read.

Use OBS=0 to create an empty data set that has the structure, but not the attributes, of another data set.

You cannot use the OBS= data set option when a WHERE statement or WHERE= data set option is specified in the same DATA or PROC step.

## Comparisons

- □ The OBS= data set option overrides the OBS= system option for the individual data set.
- □ While the OBS= data set option specifies an ending point for processing, the FIRSTOBS= data set option specifies a starting point. The two options are often used together to define a range of observations to be processed.
- □ The OBS= data set option enables you to select observations from SAS data sets. You can select observations to be read from external data files by using the OBS= option in the INFILE statement.

## Examples

In this example, the OBS= data set option in the SET statement reads in the first ten observations from data set OLD:

```
data new;
    set old(obs=10);
run;
```

This statement prints only observations 5 through 10 in data set STUDY:

```
proc print data=study(firstobs=5 obs=10);
```

## See Also

Data Set Options:
 “FIRSTOBS=” on page 18
Statements:
 “INFILE” on page 857
 “WHERE” on page 1028
System Options:
 “OBS=” on page 1129

---

## OUTREP=

**Specifies an operating environment’s requirements vector for an output file**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

### Syntax

OUTREP=*format-type*

### Syntax Description

***format-type***
specifies the requirements vector for the output file. For example, specify OUTREP=PC_ANSI if you want a file in PC format with characters that are encoded in ANSI ASCII. For more details, see *SAS/CONNECT Software: Usage and Reference* or *SAS/SHARE Software: Usage and Reference*.

---

## POINTOBS=

**Controls whether a compressed data set may be processed with random access (by observation number) rather than sequential access only**

**Valid in:** DATA step and PROC steps

**Category:** Observation Control

**Restriction:** POINTOBS= is effective only when creating a compressed data set; otherwise it is ignored.

### Syntax

POINTOBS= YES | NO

## Syntax Description

**YES**

causes SAS software to produce a compressed data set that may be randomly accessed by observation number. This is the default.

Examples of accessing data directly by observation number are:

- the POINT= option of the MODIFY and SET statements in the DATA step
- going directly to a specific observation number with PROC FSEDIT.

**Tip:** Specifying POINTOBS=YES does not affect the efficiency of retrieving information from a data set, but it does increase CPU usage by roughly 10% when creating a compressed data set and when updating or adding information to it.

**NO**

suppresses the ability to randomly access observations in a compressed data set by observation number.

**Tip:** Specifying POINTOBS=NO is desirable for applications where the ability to point directly to an observaton by number within a compressed data set is not important.

If you do not need to access data by observation number, then you can improve performance by roughly 10% when creating a compressed data set and when updating or adding observations to it by specifying POINTOBS=NO.

## Details

Note that REUSE=YES takes precedence over POINTOBS=YES. For example:

```
data test(compress=yes pointobs=yes reuse=yes);
```

results in a data set that has POINTOBS=NO. Because POINTOBS=YES is the default when you use compression, REUSE=YES causes POINTOBS= to change to NO.

## See Also

Data Set Options:

“COMPRESS= ”on page 12

“REUSE=” on page 38

System Options:

“COMPRESS=” on page 1077

“REUSE=” on page 1143

## PW=

**Assigns a read, write, or alter password to a SAS file and enables access to a password-protected SAS file**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

### Syntax

PW=*password*

### Syntax Description

***password***
must be a SAS name.

### Details

The PW= option applies to all types of SAS files except catalogs. You can use this option to assign a password to a SAS file or to access a password-protected SAS file.

### See Also

Data Set Options:
“ALTER=” on page 9
“ENCRYPT=” on page 16
“READ=” on page 34
“WRITE=” on page 46
“Manipulating Passwords” in “The DATASETS Procedure” in *SAS Procedures Guide*
“File Protection” in *SAS Language Reference: Concepts*

## PWREQ=

**Controls the pop up of a requestor window for a data set password**

**Valid in:** DATA and PROC steps
**Category:** Data Set Control

### Syntax

PWREQ=YES | NO

### Syntax Description

**YES**
specifies that a requestor window appear.

**NO**
prevents a requestor window from appearing. If a missing or invalid password is entered, the data set is not opened and an error message is written to the SAS log.

### Details

In an interactive SAS session, the PWREQ= option controls whether a requestor window appears after a user enters an incorrect or a missing password for a SAS data set that is password protected. PWREQ= applies to data sets with read, write, or alter passwords. PWREQ= is most useful in SCL applications.

### See Also

Data Set Options:

- “ALTER=” on page 9
- “ENCRYPT=” on page 16
- “PW=” on page 32
- “READ=” on page 34
- “WRITE=” on page 46

## READ=

**Assigns a read password to a SAS file and enables access to a read-protected SAS file**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

### Syntax

READ=*read-password*

### Syntax Description

***read-password***
must be a SAS name.

### Details

The READ= option applies to all types of SAS files except catalogs. You can use this option to assign a *read-password* to a SAS file or to access a read-protected SAS file.

## See Also

Data Set Options:

  "ALTER=" on page 9
  "ENCRYPT=" on page 16
  "PW=" on page 32
  "WRITE=" on page 46

"Manipulating Passwords" in "The DATASETS Procedure" in *SAS Procedures Guide*
"File Protection" in *SAS Language Reference: Concepts*

---

## RENAME=

**Changes the name of a variable**

**Valid in:** DATA step and PROC steps
**Category:** Variable Control

### Syntax

RENAME=(*old-name-1*=*new-name-1* < . . . *old-name-n*=*new-name-n*>)

### Syntax Description

***old-name***
the variable you want to rename.

***new-name***
the new name of the variable. It must be a valid SAS name.

### Details

If you use the RENAME= data set option when you create a data set, the new variable name is included in the output data set. If you use RENAME= on an input data set, the new name is used in DATA step programming statements.

If you use RENAME= on an input data set that is used in a SAS procedure, SAS changes the name of the variable in that procedure. The list of variables to rename must be enclosed in parentheses:

```
proc print data=test(rename=(score1=score2));
```

If you use RENAME= in the same DATA step with either the DROP= or the KEEP= data set option, the DROP= and the KEEP= data set options are applied before RENAME=. Thus, use *old-name* in the DROP= and KEEP= data set options. You cannot drop and rename the same variable in the same statement.

### Comparisons

- The RENAME= data set option differs from the RENAME statement in the following ways:

  - □ The RENAME= data set option can be used in PROC steps and the RENAME statement cannot.
  - □ The RENAME statement applies to all output data sets. If you want to rename different variables in different data sets, you must use the RENAME= data set option.
  - □ To rename variables before processing begins, you must use a RENAME= data set option on the input data set or data sets.
- □ Use the RENAME statement or the RENAME= data set option when program logic requires that you rename variables, for example, if two input data sets have variables with the same name. To rename variables as a file management task, use the DATASETS procedure.

## Examples

**Example 1: Renaming a Variable at Time of Output** This example uses RENAME= in the DATA statement to show that the variable is renamed at the time it is written to the output data set. The variable keeps its original name, X, during the DATA step processing:

```
data two(rename=(x=keys));
   set one;
   z=x+y;
run;
```

**Example 2: Renaming a Variable at Time of Input** This example renames variable X to a variable named KEYS in the SET statement, which is a rename before DATA step processing. KEYS, not X, is the name to use for the variable for DATA step processing.

```
data three;
   set one(rename=(x=keys));
   z=keys+y;
run;
```

## See Also

Data Set Options:

"DROP=" on page 15

"KEEP=" on page 27

Statements:

"RENAME" on page 995

"The DATASETS Procedure" in the *SAS Procedures Guide*

## REPEMPTY=

**Controls replacement of like-named temporary or permanent SAS data sets when the new one is empty.**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

**Restriction:** Use with output data sets only.

## Syntax

REPEMPTY=YES | NO

## Syntax Description

**YES**
specifies that a new empty data set with a given name replaces an existing data set with the same name. This is the default.

**Interaction:** When REPEMPTY=YES and REPLACE=NO, then the data set is not replaced.

**NO**
specifies that a new empty data set with a given name does not replace an existing data set with the same name.

**Tip:** Use REPEMPTY=NO to prevent the following syntax error from replacing the existing data set B with the new empty data set B that is created by mistake:

```
data mylib.a set b;
```

**Tip:** For both the convenience of replacing existing data sets with new ones that contain data and the protection of not overwriting existing data sets with new empty ones that are created by accident, set REPLACE=YES and REPEMPTY=NO.

## Comparisons

- For an individual data set, the REPEMPTY= data set option overrides the REPEMPTY= option in the LIBNAME statement.
- The REPEMPTY= and REPLACE= data set options apply to both permanent and temporary SAS data sets. The REPLACE system option, however, only applies to permanent SAS data sets.

## See Also

Data Set Options:
  "REPLACE=" on page 37
Statement Options:
  REPEMPTY= in the LIBNAME statement on page 915
  System Options:
    "REPLACE" on page 1142

## REPLACE=

**Controls replacement of like-named temporary or permanent SAS data sets**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

**Restriction:** Use with output data sets only.

## Syntax

REPLACE=NO | YES

## Syntax Description

**NO**
specifies that a new data set with a given name does not replace an existing data set with the same name.

**YES**
specifies that a new data set with a given name replaces an existing data set with the same name.

## Comparisons

- The REPLACE= data set option overrides the REPLACE system option for the individual data set.
- The REPLACE system option only applies to permanent SAS data sets.

## Examples

Using the REPLACE= data set option in this DATA statement prevents SAS from replacing a permanent SAS data set named ONE in a library referenced by MYLIB:

```
data mylib.one(replace=no);
```

SAS writes a message in the log that tells you that the file has not been replaced.

## See Also

System Options:
"REPLACE" on page 1142

## REUSE=

**Specifies whether new observations are written to free space in compressed SAS data sets**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

**Restriction:** Use with output data sets only.

## Syntax

REUSE=NO | YES

### Syntax Description

**NO**
: does not track and reuse space in compressed data sets. New observations are appended to the existing data set. Specifying the NO argument results in less efficient data storage if you delete or update many observations in the SAS data set.

**YES**
: tracks and reuses space in compressed SAS data sets. New observations are inserted in the space that is freed when other observations are updated or deleted.

  If you plan to use procedures that add observations to the end of SAS data sets (for example, the APPEND and FSEDIT procedures) with compressed data sets, use the REUSE=NO argument. REUSE=YES causes new observations to be added wherever there is space in the file, not necessarily at the end of the file.

### Details

By default, new observations are appended to existing compressed data sets. If you want to track and reuse free space by deleting or updating other observations, use the REUSE= data set option when you create a compressed SAS data set.

REUSE= has meaning only when you are creating new data sets with the COMPRESS=YES data set option or system option. Using the REUSE= data set option when you are accessing an existing SAS data set has no effect.

### Comparisons

The REUSE= data set option overrides the REUSE= system option.

Note that REUSE=YES takes precedence over POINTOBS=YES. For example:

```
data test(compress=yes pointobs=yes reuse=yes);
```

results in a data set that has POINTOBS=NO. Because POINTOBS=YES is the default when you use compression, REUSE=YES causes POINTOBS= to change to NO.

### See Also

Data Set Options:
    “COMPRESS= ”on page 12
System Options:
    “REUSE=” on page 1143

## SORTEDBY=

**Specifies how the data set is currently sorted**

**Valid in:** DATA step and PROC steps
**Category:** Data Set Control

### Syntax

SORTEDBY=*by-clause* <*/ collate-name*> | _NULL_

## Syntax Description

***by-clause < / collate-name>***
indicates how the data are currently sorted.

| | |
|---|---|
| *by-clause* | are the variables and options that you use in a BY statement in a PROC SORT step. |
| *collate-name* | names the collating sequence that is used for the sort. By default, the collating sequence is that of your operating environment. A slash (/) must precede the collating sequence. *Operating Environment Information:* For details on collating sequences, see the SAS documentation for your operating environment. △ |

**_NULL_**
removes any existing sort information.

## Details

SAS uses the sort information in these ways:

- For BY-group processing, if the data are already sorted by the BY variable, SAS does not use the index, even if the data set is indexed on the BY variable.
- If an index is selected for WHERE expression processing, the sort information for that data set is changed to reflect the order that is specified by the index.
- At the time you create an index, the sort information can make sorting of key variables unnecessary.
- PROC SQL uses the sort information to process queries more efficiently and to determine whether an internal sort is necessary before performing a join.
- PROC SORT checks for the sort information before it sorts a data set so that data are not resorted unnecessarily.
- PROC SORT sets the sort information whenever it does a sort.

If you update a SAS file in a way that affects the validity of the sort, the sort information is removed. That is, if you change or add any values of the variables by which the data set is sorted, the sort information is removed.

## Comparisons

- Use the CONTENTS statement in the DATASETS procedure to see how a data set is sorted.
- The SORTEDBY= option does not cause a data set to be sorted.

## Examples

This example uses the SORTEDBY= data set option to specify how the data are currently sorted. The data set ORDERS is sorted by PRIORITY and by the descending values of INDATE. Once the data set is created, the sort information is stored with it. These statements create the data set ORDERS and record the sort information:

```
libname mylib 'SAS-data-library';
options yearcutoff=1920;
```

```
data mylib.orders(sortedby=priority
                  descending indate);
   input priority 1. +1 indate date7.
         +1 office $ code $;
   format indate date7.;
   datalines;
1 03may01 CH J8U
1 21mar01 LA M91
1 01dec00 FW L6R
1 27feb99 FW Q2A
2 15jan00 FW I9U
2 09jul99 CH P3Q
3 08apr99 CH H5T
3 31jan99 FW D2W
;
```

### See Also

The CONTENTS Statement in "The DATASETS Procedure" in the *SAS Procedures Guide*

"The SORT Procedure" in the *SAS Procedures Guide*

"The SQL Procedure" in the *SAS Procedures Guide*

## TOBSNO=

**Specifies the number of observations to be transmitted in each multi-observation exchange with a SAS server**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

**Restriction:** The TOBSNO= option is valid only for data sets that are accessed through a SAS server via the REMOTE engine.

**See:** The description of the FOPEN function in *SAS Component Language: Reference*

### Syntax

TOBSNO=*n*

### Syntax Description

***n***

specifies the number of observations to be transmitted.

## TRANTAB=

**Specifies a translation table for character conversions**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

### Syntax

TRANTAB=*table-name*

### Syntax Description

***table-name***
specifies the name of a translation table that is used to translate the characters in a data set from foreign encoding to local encoding. SAS searches for translation tables in SASUSER.PROFILE first and then SASUSER.HOST.

*Note:* The TRANTAB= data set option overrides the libname option for a specific file. △

### Details

Translation tables are used internally by the SAS supervisor to implement National Language Support (NLS). If you are unfamiliar with the purpose of translation tables, do not change the specifications without proper technical advice. For details, see *SAS/CONNECT Software: Usage and Reference* or *SAS/SHARE Software: Usage and Reference.*

## TYPE=

**Specifies the data set type for a specially structured SAS data set**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

### Syntax

TYPE=*data-set-type*

### Syntax Description

***data-set-type***
specifies the special type of the data set.

### Details

Use the TYPE= data set option in a DATA step to create a special SAS data set in the proper format, or to identify the special type of the SAS data set in a procedure statement.

You can use the CONTENTS procedure to determine the type of a data set.

Most SAS data sets do not have a specified type. However, there are several specially structured SAS data sets that are used by some SAS/STAT procedures. These SAS data sets contain special variables and observations, and they are usually created by SAS statistical procedures. Because most of the special SAS data sets are used with SAS/STAT software, they are described in the *SAS/STAT User's Guide*.

Other values are available in other SAS software products and are described in the appropriate documentation.

*Note:* If you use a DATA step with a SET statement to modify a special SAS data set, you must specify the TYPE= option in the DATA statement. The *data-set-type* is not automatically copied to the data set that is created. △

### See Also

"Special SAS Data Sets" in the *SAS/STAT User's Guide*

"The CONTENTS Procedure" in the *SAS Procedures Guide*

## WHERE=

**Selects observations that meet the specified condition**

**Valid in:** DATA step and PROC steps

**Category:** Observation Control

**Restriction:** Cannot be used with the FIRSTOBS= and OBS= data set options or with the POINT= option in the SET and MODIFY statements.

### Syntax

WHERE=(*where-expression*)

### Syntax Description

***where-expression***
is an arithmetic or logical expression that consists of a sequence of operators, operands, and SAS functions. The expression must be enclosed in parentheses.

### Details

- Use the WHERE= data set option with an input data set to select observations that meet the condition specified in the WHERE expression before SAS brings them into the DATA or PROC step for processing. Selecting observations that meet the conditions of the WHERE expression is the first operation SAS performs in each iteration of the DATA step.

You can also select observations that are written to an output data set. In general, selecting observations at the point of input is more efficient than selecting them at the point of output; however, there are some cases when selecting observations at the point of input is not practical or not possible.

- □ You cannot use the WHERE= data set option with options that select observations by observation number, such as the FIRSTOBS= and OBS= data set options and the POINT= option in the SET and MODIFY statements.
- □ If you use both the WHERE= data set option and the WHERE statement in the same DATA step, SAS ignores the WHERE statement for data sets with the WHERE= data set option. However, you can use the WHERE= data set option with the WHERE command in SAS/FSP software.

*Note:* Using indexed SAS data sets can improve performance significantly when you are using WHERE expressions to access a subset of the observations in a SAS data set. See “SAS Indexes” in “SAS Data Files” in *SAS Language Reference: Concepts* for a complete discussion of WHERE expression processing with indexed data sets and a list of guidelines to consider before indexing your SAS data sets. △

## Comparisons

- □ The WHERE statement applies to all input data sets, whereas the WHERE= data set option selects observations only from the data set for which it is specified.
- □ Do not confuse the purpose of the WHERE= data set option. The DROP= and KEEP= data set options select variables for processing, while the WHERE= data set option selects observations.

## Examples

**Example 1: Selecting Observations from an Input Data Set** This example uses the WHERE= data set option to subset the SALES data set as it is read into another data set:

```
data whizmo;
   set sales(where=(product='whizmo'));
run;
```

**Example 2: Selecting Observations from an Output Data Set** This example uses the WHERE= data set option to subset the SALES output data set:

```
data whizmo(where=(product='whizmo'));
   set sales;
run;
```

## See Also

Statements:
 "WHERE" on page 1028
"WHERE Processing" in *SAS Language Reference: Concepts*

## WHEREUP=

**Specifies whether to evaluate added observations and modified observations against a WHERE expression**

**Valid in:** DATA step and PROC steps

**Category:** Observation Control

### Syntax

WHEREUP= NO | YES

### Syntax Description

**NO**
does not evaluate added observations and modified observations against a WHERE expression.

**YES**
evaluates added observations and modified observations against a WHERE expression.

### Details

Specify WHEREUP=YES when you want any added observations or modified observations to match a specified WHERE expression.

### Examples

**Example 1: Accepting Updates That Do Not Match the WHERE expression** This example shows how WHEREUP= permits observations to be updated and added even though the modified observation does not match the WHERE expression:

```
data a;
   x=1;
   output;
   x=2;
   output;
run;

data a;
   modify a(where=(x=1) whereup=no);
   x=3;
```

```
      replace; /* Update does not match WHERE expression */
      output; /* Add does not match WHERE expression */
   run;
```

In this example, SAS updates the observation and adds the new observation to the data set.

**Example 2: Rejecting Updates That Do Not Match the WHERE expression** In this example, WHEREUP= does not permit observations to be updated or added when the update and the add do not match the WHERE expression:

```
   data a;
      x=1;
      output;
      x=2;
      output;
   run;

   data a;
      modify a(where=(x=1) whereup=yes);
      x=3;
      replace; /* Update does not match WHERE expression */
      output; /* Add does not match WHERE expression */
   run;
```

In this example, SAS does not update the observation nor does it add the new observation to the data set.

### See Also

Data Set Option:

“WHERE=” on page 43

## WRITE=

**Assigns a write password to a SAS file and enables access to a write-protected SAS file**

**Valid in:** DATA step and PROC steps

**Category:** Data Set Control

### Syntax

WRITE=*write-password*

### Syntax Description

***write-password***
must be a SAS name.

## Details

The WRITE= option applies to all types of SAS files except catalogs. You can use this option to assign a *write-password* to a SAS file or to access a write-protected SAS file.

## See Also

Data Set Options:

"ALTER=" on page 9

"ENCRYPT=" on page 16

"PW=" on page 32

"READ=" on page 34

"Manipulating Passwords" in "The DATASETS Procedure" in the *SAS Procedures Guide*

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Language Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999.

**SAS® Language Reference, Version 8**

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1–58025–369–5

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of the software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227–19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The Institute is a private company devoted to the support and further development of its software and related services.